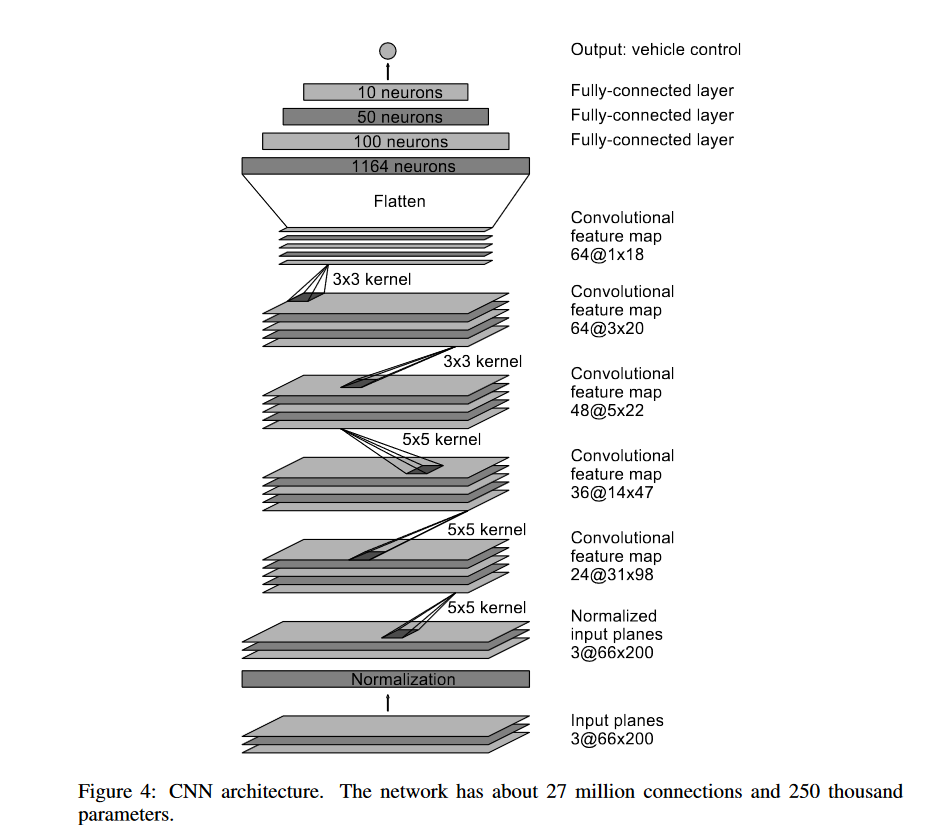
**Sunday, April 7th, 2024**

I tested out the lidar-based navigation code some more continuously. It worked decently well when I re-configured the “track”, with only a few crashes when the car went too fast. However, I did discover a bigger roadblock for reinforcement learning, or any sort of extended operation of the car in general. The ESC that I’m using overheats after a few minutes of the car running continuously. I know this because the overheating status light turns on and it is also extremely hot to the touch. Thus, if I wanted to run the car for longer, I would probably need a better solution such as a different ESC rated for more power. (This ESC is the backup one I got when I wasn’t able to effectively interface with the VESC motor controller).

Furthermore, the crashes which did occur were because the lidar data did not update fast enough to detect obstacles in front of the car. Thus, I am somewhat wary of letting the car run unsupervised for a long period of time, because of the potential for crashes. The front left corner of the wooden mounting plate is already broken due to a collision.

**Monday, April 8, 2024**

Given the constraints described above, I started looking into ways that I could possibly reduce the dimensionality of the reinforcement learning using the lidar sensor instead of by training an autoencoder on camera images. Using the lidar array of 360 values is definitely a much smaller input than an entire RGB image. However, I still need to determine a good way to avoid crashes during the early stages of the reinforcement learning. The problem is that reinforcement learning, by definition, won’t do that.

I found a paper (<https://arxiv.org/abs/1810.02890>) which uses behavioral cloning (e.g. a CNN) to bootstrap the reinforcement learning process. Thus, I took a look at the dataset I had recorded for this very purpose over spring break.

I tried applying the standard Nvidia CNN model (right). The training completed with an loss of 0.03, which was encouraging. However, I then plotted the true steering against the predicted steering by the CNN and the CNN was woefully incorrect. Here is an example of the true (blue) and predicted (green) steering and throttle. In this case, the CNN would crash into the wall (see next page).

A screenshot of a video

Description automatically generated

Most of the predictions were quite decent. However, this doesn’t quite solve the problem of preventing crashes entirely.

I was informed that I needed to start working on my poster, so I started on that. I downloaded the template from the CSL website and then found another template of the same size.

**Wednesday, April 10th,** **2024**

There was no school today. I continued working on my poster.